Learning to Prune: A Reduce-then-Optimize Approach for the Capacitated Vehicle Routing Problem

Baptiste Vert*[1], Carolin Schmidt[2], and Maximilian Schiffer[3]

[1]Department of Operations & Technology, Technical University of Munich, Germany
[2]Dr., Department of Operations & Technology, Technical University of Munich, Germany
[3]Pr. Dr., Department of Operations & Technology, Technical University of Munich, Germany

Short summary

Recent advances in mathematical programming have enabled exact solutions of the Capacitated Vehicle Routing Problem (CVRP) for hundreds of customers, while metaheuristics scale to thousands. A defining property of CVRP solutions is their sparsity: optimal solutions use only a small fraction of edges in the complete graph. We exploit this through a reduce-then-optimize pipeline in which a Graph Neural Network predicts which edges are likely to appear in the optimal solution, pruning the graph before solving. On 100-customer benchmark instances, retaining just 20% of edges yields optimal solutions within 1% of optimality while cutting runtime by up to 40%. Graph reduction further enables exact solution of 15% of test instances otherwise intractable on the complete graph. On instances with up to 2,000 customers, GNN-guided candidate lists improve metaheuristic efficiency by 40% without quality loss. The approach extends naturally to the CVRP with Time Windows, showing consistent gains across both solver types.

Keywords: Artificial Intelligence and Big Data Analytics, Operations research applications, Urban freight transport and city logistics

1 Introduction

The Capacitated Vehicle Routing (CVRP) has been a cornerstone of operations research since (Clarke & Wright, 1964), with applications in logistics, urban freight and supply chain management. Modern exact methods, most notably the branch-cut-and-price solver of (Pessoa, Sadykov, Uchoa, & Vanderbeck, 2020), can now solve instances with up to 100 customers in minutes and up to 1,000 customers within days. Yet scaling remains a fundamental challenge, and practitioners routinely rely on metaheuristics such as Hybrid Genetic Search (HGS) (Wouda, Lan, & Kool, 2024) for larger instances. The intrinsic sparsity of CVRP solutions, where only a small fraction of edges in the complete graph appears in an optimal solution, motivates a reduction-based approach: if we can identify which edges are likely to appear in an optimal solution before solving, and prune the graph accordingly, we can reduce the size of the problem an exact solver or a metaheuristic must handle, potentially unlocking significant computational gains without sacrificing solution quality.

Machine learning has recently gained traction as a tool to support combinatorial optimization (Subramanian & Teichgraeber, 2023), and can be leveraged to answer this question. While end-to-end neural solvers (Kool, van Hoof, & Welling, 2019) replace combinatorial optimization techniques with fully learned pipelines, our approach complements rather than replaces existing exact and heuristic solvers, preserving their constraint-handling guarantees and decades of tailored algorithmic development. Recent work has shown that reduce-then-optimize strategies can yield near-optimal solutions with significant runtime reductions for the Fixed-Charge Transportation Problem (Spieckermann, Minner, & Schiffer, 2025). We extend this paradigm to the CVRP, using a Graph Neural Network (GNN) that predicts the likelihood of each edge appearing in an optimal solution. These probabilities are used to (i) prune the input graph for an exact solver, and (ii) construct candidate lists for a metaheuristic. GNNs are particularly well-suited to routing problems, as they can learn structural patterns, i.e., demand distributions, that classical distance-based pruning rules (Wouda et al., 2024) cannot capture. Finally, we also explore whether this reduction strategy generalises to the more complex CVRP with Time Windows (CVRPTW).

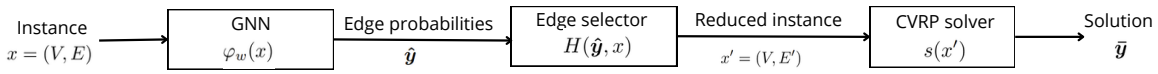


Figure 1: The reduce-then-optimize pipeline.

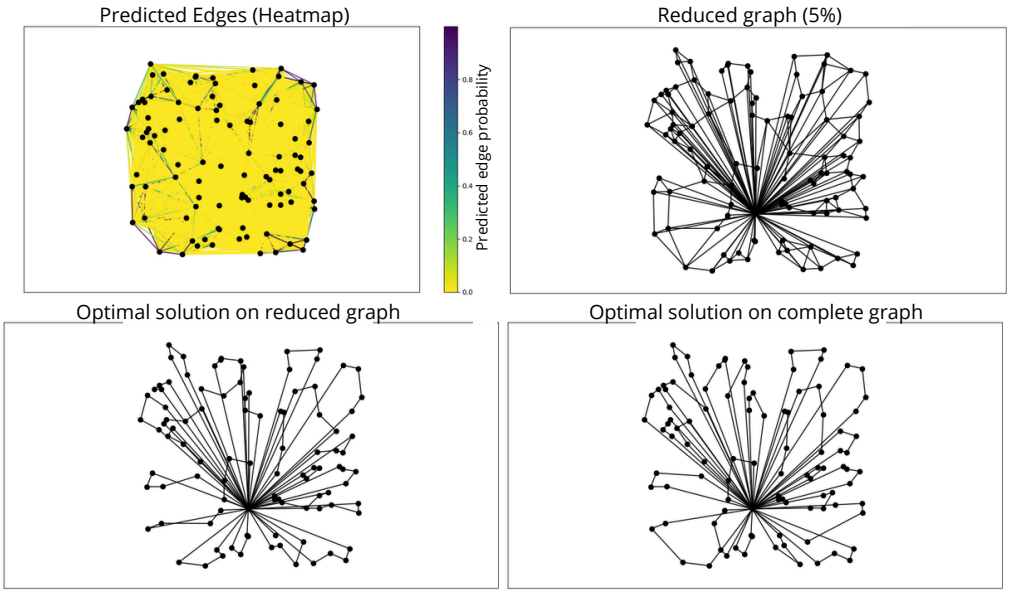


Figure 2: Graph reduction on a 100-customer instance: Reduction to 5% of the complete graph leads to an optimality gap of 2.7%.

2 Methodology

Given a CVRP instance $x = (V, E)$ with node set V and edge set E, a GNN is trained to predict the likelihood $\varphi_w(x) = \hat{\boldsymbol{y}}$ that each edge belongs to an optimal solution. At inference time, a quantile-based threshold θ is applied to retain only the top fraction of edges by predicted probability, yielding a reduced graph $x' = (V, H(\hat{\boldsymbol{y}}, x))$. This reduced graph is then passed to a solver—either as input graph to an exact solver or as a candidate list for a metaheuristic—to obtain the final solution $s(x') = \bar{\boldsymbol{y}}$. The full pipeline is illustrated in Figure 1.

Neural Network Architecture To exploit the natural graph structure of the CVRP, we employ a multi-layer GNN with a bidirectional message-passing operator, adapted from the Graph Attention Network (Veličković et al., 2018). Since the final prediction targets edges rather than nodes, we leverage an architecture that updates node and edge embeddings jointly at every layer through bidirectional message passing. Several such layers are stacked, and a final multi-layer perceptron followed by sigmoid activation is applied to the edge embeddings, which produces a score for each edge. Node features include customer demand and vehicle capacity. Edge features include the Euclidean travel distance, as well as additional structural features that include Minimum Spanning Tree (MST) membership, Voronoi adjacency, and Clarke–Wright savings values.

Training CVRP solutions are intrinsically sparse, meaning positive edges (those appearing in an optimal solution) are heavily outnumbered by negative ones. To address this severe class imbalance, we train with a weighted binary cross-entropy loss.

$$\mathcal{L}(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|E_i|}\sum_{e\in E_i} w_e^i[y_e \log(\hat{y}_e) + (1-y_e)\log(1-\hat{y}_e)] \tag{1}$$

where N denotes the number of instances and E_i corresponds to the set of edges of instance i, and class weight w_e is defined as:

$$w_e = \begin{cases} \frac{\sum_{i\in[\![1,N]\!]}\sum_{e\in E_i}\mathbf{1}_{y_e=1}}{\sum_{n\in[\![1,N]\!]}\sum_{e\in E_k}\mathbf{1}_{y_e=0}} & \text{if } y_e = 1 \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

which upweights positive edges in proportion to the overall class imbalance. Once trained, the GNN assigns a probability $\hat{y}_{i,j}$ to each edge. A quantile-based threshold θ, referred as size-threshold, is then applied to retain a target fraction of edges, yielding a reduced graph.

To guarantee feasibility of the reduced graph, we apply a post-processing step before invoking the solver. Specifically, we use HGS via PyVRP (Wouda et al., 2024) with modified edge costs:

$$c_{i,j} = \begin{cases} \dfrac{1}{\hat{y}_{i,j} + \epsilon}, & \text{if } \hat{y}_{i,j} > \theta \\ \dfrac{100}{\hat{y}_{i,j} + \epsilon}, & \text{otherwise} \end{cases} \tag{3}$$

where $\epsilon > 0$ is a small constant for numerical stability. This cost structure encourages the initial solution to use predicted edges while still allowing it to incorporate unpredicted edges if feasibility requires it. In case unpredicted edges are added, we add them to the reduced graph.

Experimental Setup We evaluate on four dataset variants. For small CVRP instances, we use a benchmark (Queiroga, Sadykov, Uchoa, & Vidal, 2022) with 100 customers, split into 8,000 training, 1,000 validation, and 501 test instances. For large CVRP instances, we follow the generation procedure of (Uchoa et al., 2017) across five size categories (500, 750, 1000, 1500, and 2000 customers) with 400 training, 50 validation, and 50 test instances per size.

The CVRPTW variants are constructed by augmenting both instance sets with time windows, generated by sampling a window width uniformly between 20% and 30% of the time horizon H, defined as :

$$H = 2 \max t(0, v) + \left\lceil \frac{\sum d_i}{Q} \right\rceil \left(\frac{\sum t(u, v)}{|A|} + S \right), \tag{4}$$

where $t(0, v)$ denotes the travel time from the depot to client v, Q denotes the capacity of the vehicle, $|A|$ denotes the number of arcs of the graph and S denotes the service time. We replaced the distances by durations, using a fixed speed. Of the 1,000 small CVRPTW instances generated this way, 940 were feasible, as the time windows were chosen tightly on purpose to create solutions where time windows would have an impact on the form of the solution.

For the labels of the small CVRP instances, we use the optimal solutions provided by the CVRPLIB initiated by (Uchoa et al., 2017). For the bigger CVRP, small and big CVRPTW instances, we used PyVRP (Wouda et al., 2024) to compute the best-known solutions within a time limit of 1000s.

In our reduce-then-optimize pipeline, for small instances, the reduced graph is passed to the exact solver (Pessoa et al., 2020) through the VRPSolverEasy API (Errami, Queiroga, Sadykov, & Uchoa, 2023), with a time limit of 5000 s. For large instances, exact solving is computationally intractable; instead HGS via PyVRP (Wouda et al., 2024) is run on a reduced graph and the local search candidate list for each customer is restricted to the top k node pairs according to their predicted edge probability. For the CVRPTW instances, unpredicted arcs are penalized with large travel durations, such that they are not considered in the candidate list. Evaluation is run i9-9900 @ 16x3.10 GHz for small instances and on Intel(R) Xeon(R) Platinum 8380 for large instances.

3 Results and discussion

We evaluate the pipeline in two stages: first, the predictive performance of the GNN as a classifier, then the downstream impact of graph reduction on solution quality and computational runtime.

Edge-prediction performance is evaluated using standard classification metrics, i.e., accuracy, recall, and precision. Downstream impact on solution quality and runtime, is reported using metrics relative to the computation on the complete graph. The *Relative Value Gap* (RVG) measures solution cost degradation against the known best cost, while the *Relative Runtime Gap* (RRG) captures the change in total computational time, including feature computation, inference, feasibility completion, and solver time:

$$\text{RRG} = \frac{ReduRunt - CompRunt}{CompRunt} \tag{5}$$

$$\text{RVG} = \frac{ReduCost - BestCost}{BestCost} \tag{6}$$

Small instances (100 customers)

Table 1 reports GNN performance on the test set at a decision threshold of $\hat{y}_{i,j} > 0.5$. The model achieves a recall of 96.2%, demonstrating that the majority of optimal edges are retained despite severe class imbalance. The relatively low precision (24.1%) reflects the expected trade-off, that the model retains more edges to avoid discarding optimal ones.

Table 1: GNN performance on the test set.

Metric	**Value(%)**
Precision	24.1
Accuracy	92.9
Recall	96.2

Table 2 shows how recall varies as the size-threshold decreases. Retaining one third of the edges yields a recall of 99.96%, meaning virtually no optimal edge is lost on average. Even at a threshold of 0.1, recall remains above 97%.

Table 2: GNN recall on the test set.

Size-Threshold	0.3333	0.2666	0.2	0.1	0.05	0.02
Recall (%)	99.96	99.89	99.67	97.06	87.82	59.89

As a baseline comparison, Table 3 reports the recall of a k-nearest-neighbor (k-NN) reduction, an approach for constructing candidate edge lists in metaheuristics (Wouda et al., 2024). It retains edges based purely on distance. The GNN achieves more than 8 percentage points higher recall at equivalent graph size (99.96% vs. 91.50% at one third of the edges), and consistently outperforms across all thresholds.

Table 3: k-NN recall on the test set.

k	30	20	16	8	4	2
Recall (%)	91.50	88.11	86.51	80.68	71.90	58.60

Figure 3 reports RVG and RRG across size-thresholds and probability-thresholds. For the RRG, the top 1% of runtime values are excluded as outliers, as these correspond to numerically unstable solver runs. Retaining 20% of edges is sufficient to recover optimal solutions for the majority of instances with a 25% reduction in median runtime. At 10%, solutions remain near-optimal with an average gap below 1%. Only at 5% does solution quality begin to degrade noticeably, with an average gap below 10%, which still competitive given the 95% reduction in graph size. The large standard deviation values for RRG are mainly due to well-known numerical instability of the exact solver on varying instance structures.

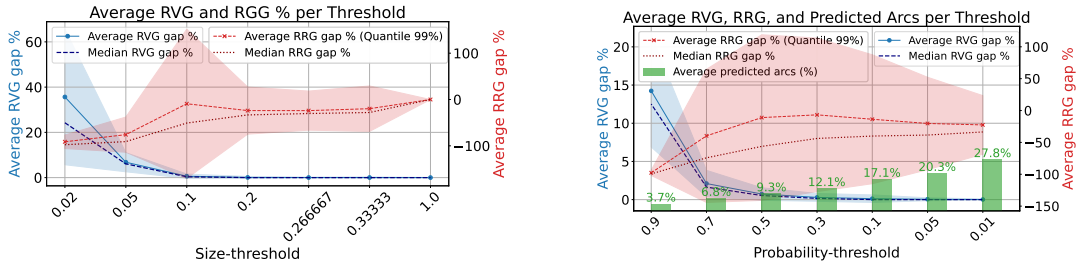


Figure 3: RVG and RRG across size-thresholds (left) and probability-thresholds (right) with standard deviation bands.

Figure 4 shows the full distribution of RVG and RRG values across all retained instances as histograms. The RVG distributions are strongly concentrated near zero for thresholds of 0.1 and above, while RRG concentrates below zero, confirming that near-optimality holds across the test set. Crucially, 76 instances, which could not be solved to optimality on the complete graph within the time limit were successfully solved on the reduced graph. At a size threshold of 20%, 43 of these instances were solved with an average optimality gap below 0.015% and an average runtime of 2,000 seconds, less than half the time limit. This suggests that graph reduction does not merely accelerate solving for tractable instances, but can also unlock solutions for instances that are otherwise computationally intractable on the full graph.

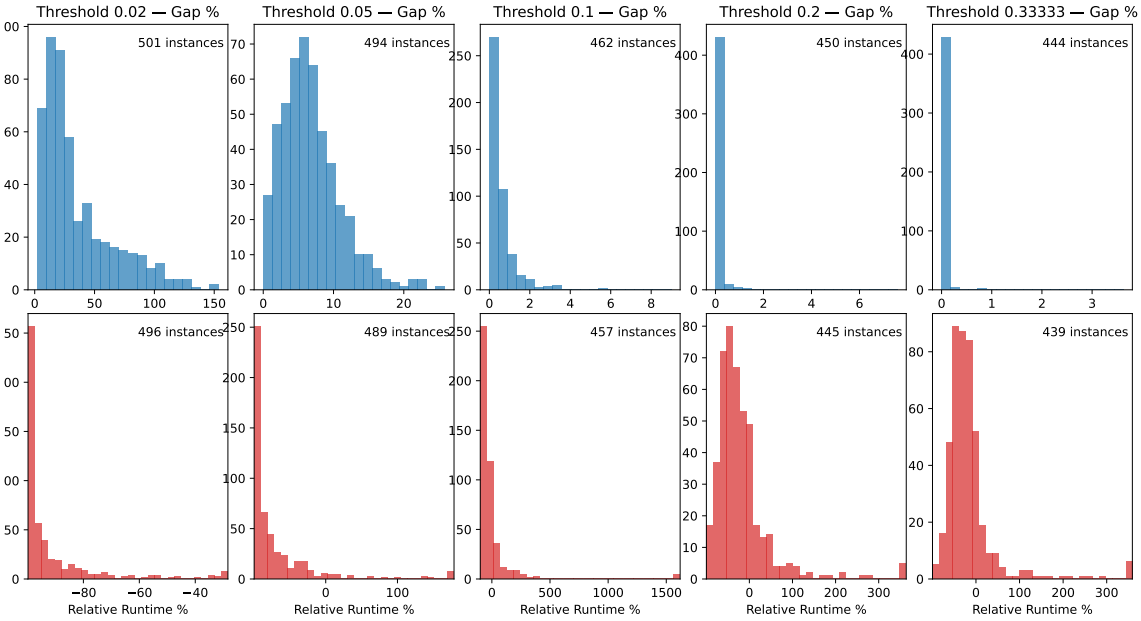


Figure 4: Distribution of RVG (top) and RRG (bottom) across retained test instances at each size-threshold. Instance count indicates the number of instances solved within the time limit at that threshold.

Figure 5 extends the comparison to the baseline k-NN method to solution quality and runtime. We can see that GNN-reduced graphs yield lower average RVG and RRG than k-NN reductions at equivalent sizes, with notably smaller variance. This stability advantage is practically significant, as the k-NN reduction produces occasional large gaps that the GNN avoids.

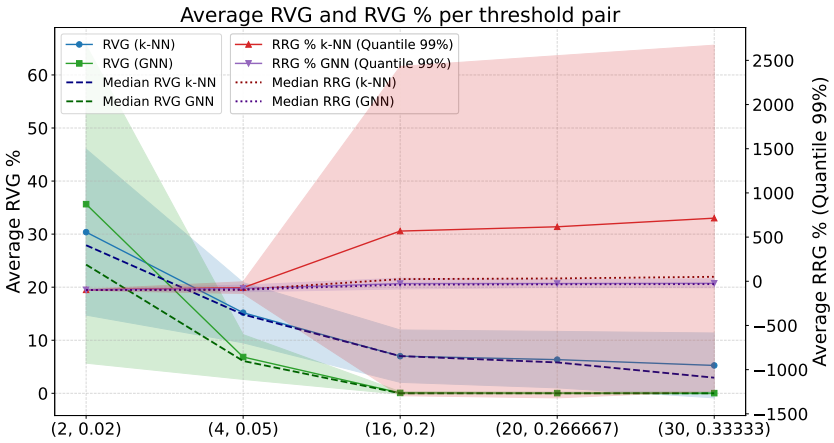


Figure 5: Mean RVG and RRG for GNN-based and k-NN-based graph reduction. X-axis labels indicate (k, threshold) pairs representing the number of neighbors per node in k-NN and the corresponding size threshold in the reduce-then-optimize approach.

Large Instances (500-2,000 customers)

For instances with more than 1,000 customers, exact solving is intractable and we instead evaluate the effect of GNN-guided candidate list restriction on PyVRP.

Table 4 reports classification performance on the large-instance test set, where we discard the additional structural edge features due to prohibitive computational runtime. Results show that recall remains high at 97.0%, though precision drops to 4.8% due to the far greater number of non-optimal edges at this scale. As shown in Table 5, recall degrades less pronounced than for small instances, as retaining just 5% of edges still yields a 97.12% recall. This observation is consistent with the intuition that larger graphs can be reduced more aggressively while preserving solution-relevant structure.

Table 4: Test set performance.

Metric	**Value (%)**
Precision	4.8
Accuracy	95.5
Recall	97.0

Table 5: GNN recall on the test set.

Size-Threshold	0.3333	0.2666	0.2	0.1	0.05	0.02	0.01	0.005
Recall (%)	99.97	99.94	99.84	99.17	97.12	89.57	76.63	56.57

Figure 6 reports RVG and RRG for HGS across restriction configurations, evaluated with a 60-second time limit. Here, RVG is calculated relative to the best objective value found by unrestricted HGS on the complete graph at the end of the full time limit, while RRG is calculated relative to the time needed to reach that best solution. The feasibility-repair phase was removed for the larger instances because the GNN consistently produced restricted graphs that were already feasible.

As observed in Figure 6, runtime drops by roughly 40% when using candidate lists of size 15–20, but very small lists fail to reach solutions as good as those obtained on the full graph within the 60-second limit. This explains why no negative RVG appears for the smallest sizes: the runtime savings on some instances are not enough to offset the poorer solution quality observed on others.

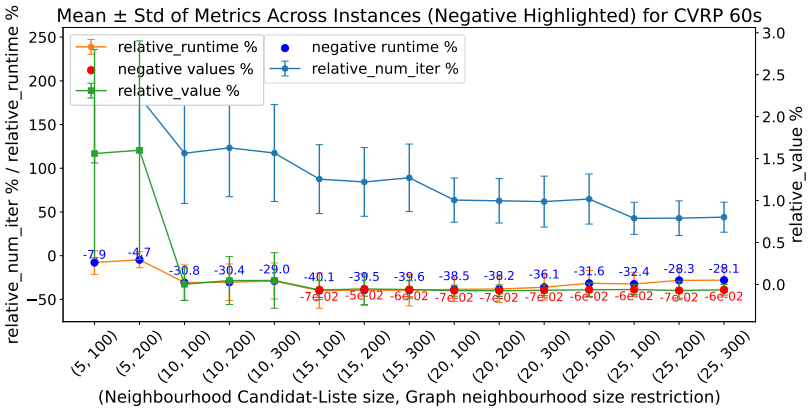


Figure 6: Mean relative runtime and solution value across restriction configurations for large CVRP instances with runtime limit of 60s. Each pair on the x-axis labels denotes (candidate list neighborhood size, the global graph restriction size).

Figure 7 illustrates the solving process of HGS on the complete graph against the reduce-then-optimize configurations with a candidate list neighborhood size of 20. All restricted variants start from a lower initial solution value and converge faster than the unrestricted baseline, confirming that focusing the local search on high-probability edges allows the metaheuristic to explore a smaller but higher-quality neighbourhood at each iteration, leading to more improvement per unit of computation time. Crucially, the convergence curves of the restricted variants closely approach the final solution value of the unrestricted PyVRP within a fraction of the total runtime, with the same or even better asymptotic solution quality.

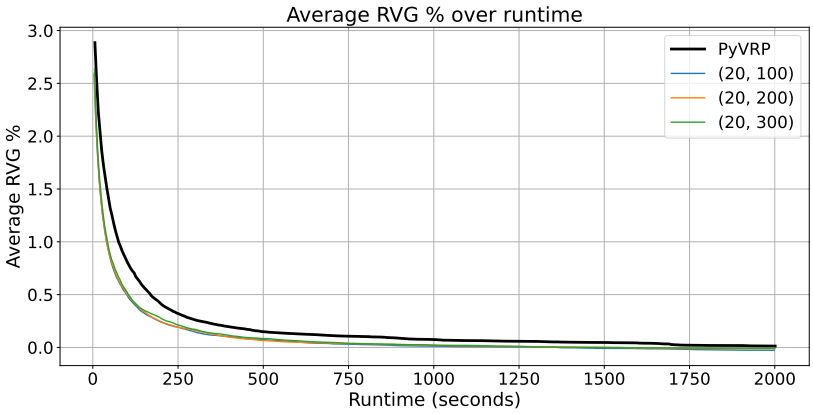


Figure 7: Average RVG [%] over runtime for PyVRP on large CVRP instances across restriction configurations.

CVRP with Time Windows

We further evaluate our approach on CVRPTW instances, where the directed graph structure and time-feasibility constraints make the problem substantially harder. Table 6 reports GNN recall over thresholds on the test set. The recall pattern closely mirrors the CVRP results, missing no optimal edge down to a size threshold of 0.2. Figure 8 confirms that RVG and RRG distributions are broadly comparable to those observed for CVRP, with near-optimal solutions recovered at moderate thresholds and meaningful runtime reductions across the test set.

Table 6: GNN recall on the test set containing small instances.

Size-Threshold	0.3333	0.2666	0.2	0.1	0.05	0.02
Recall (%)	100.00	100.00	100.00	97.14	82.86	55.23

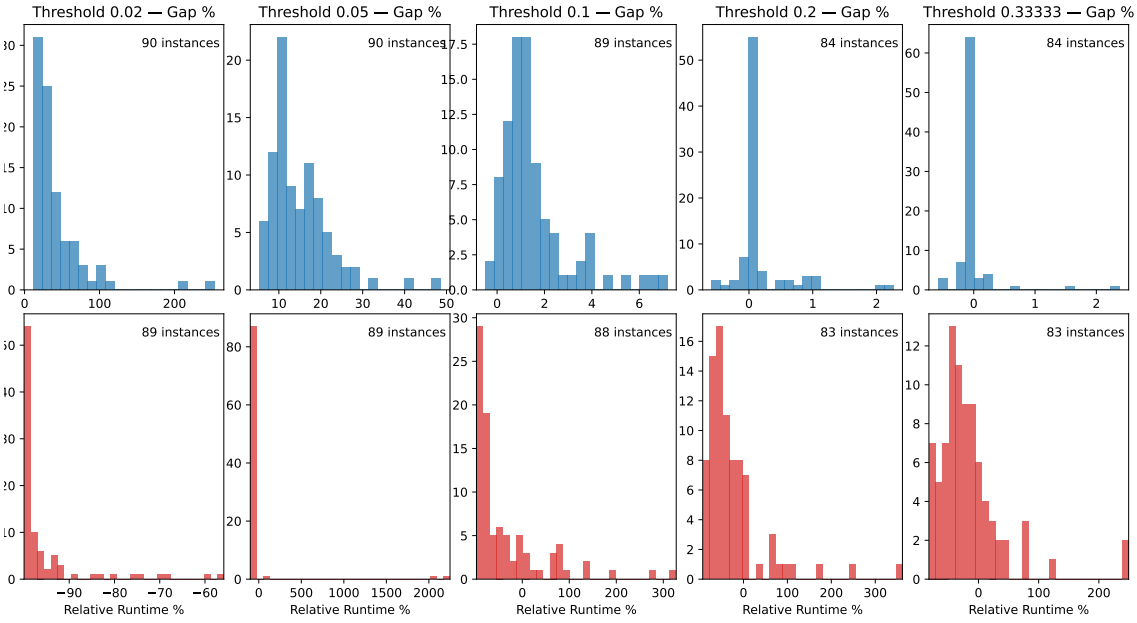


Figure 8: Distribution of RVG (top) and RRG (bottom) across retained small CVRPTW test instances at each size-threshold.

Table 7 shows that recall remains high across thresholds for even for large CVRPTW instances, and that larger graphs can be reduced more aggressively while maintaining high recall, which is consistent with the behavior observed for CVRP instances. Figure 9 shows that GNN-guided candidate list restriction yields runtime reductions of up to 36% without degrading final solution quality. Together, these results suggest that the reduce-then-optimize pipeline transfers directly to more complex routing variants without any fundamental change to the methodology.

Table 7: GNN recall on the test set for CVRPTW containing large instances.

Size-Threshold	0.3333	0.2666	0.2	0.1	0.05	0.02	0.01	0.005
Recall (%)	99.93	99.88	99.75	98.84	96.38	87.57	74.93	57.56

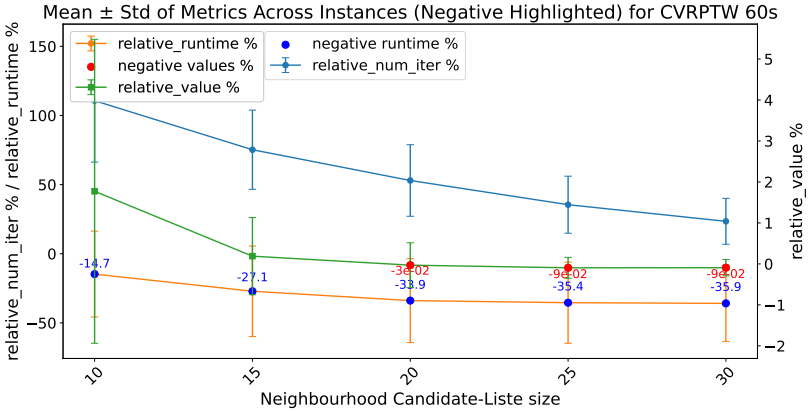


Figure 9: Mean relative runtime and solution value across restriction configurations for large CVRPTW instances with a runtime limit of 60s.

4 Conclusions

We have presented a reduce-then-optimize pipeline for the CVRP in which a GNN predicts edge membership in the optimal solution, which allows us to significantly prune the graph prior to solving. On 100-customer benchmark instances, retaining just 10% of the graph yields near-optimal solutions with an average optimality gap below 1%, while reducing solver runtime by 40% on average. Crucially, graph reduction not only accelerates solving, but enables exact solution of approximately 15% of test instances that remain intractable on the complete graph within the same time limit. On large-scale instances with up to 2,000 customers, GNN-guided candidate list restriction and graph pruning enable HGS to reach equivalent solution quality up to 40% faster. We further show that our approach generalizes to the more complex CVRP with Time Windows, where the directed graph and time-feasibility make the problem substantially harder, with consistent gains observed across both exact and heuristic solvers. A current limitation is that the GNN is trained on instances drawn from specific benchmark distributions, and generalization to out-of-distribution instances has not been fully characterized. Future work includes developing adaptive per-instance thresholds rather than a fixed-size threshold applied uniformly. More broadly, this work suggests that learned graph reduction is a promising building block for scaling exact and heuristic methods to routing problems of industrial relevance.

Acknowledgements

„Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – TRR: 408/1 2025 – Project number: 531327426“.

References

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581.

Errami, N., Queiroga, E., Sadykov, R., & Uchoa, E. (2023). Vrpsolvereasy: a python library for the exact solution of a rich vehicle routing problem. *INFORMS Journal on Computing*, *36*(4), 956–965.

Kool, W., van Hoof, H., & Welling, M. (2019). Attention, learn to solve routing problems! In *International conference on learning representations.*

Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2020). A generic exact solver for vehicle routing and related problems. *Mathematical Programming*, *183*(1), 483–523.

Queiroga, E., Sadykov, R., Uchoa, E., & Vidal, T. (2022). 10,000 optimal CVRP solutions for testing machine learning based heuristics. In *Aaai-22 workshop on machine learning for operations research (ml4or).*

Spieckermann, C., Minner, S., & Schiffer, M. (2025). Reduce-then-optimize for the fixed-charge transportation problem. *Transportation Science*, *59*(3), 540-564.

Subramanian, A., & Teichgraeber, H. (2023). The interplay between operations research and machine learning. *OR/MS Today*, *50*(2), –. doi: 10.1287/orms.2023.02.02

Uchoa, E., Pecin, D., Pessoa, A., Poggi, M., Vidal, T., & Subramanian, A. (2017). New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, *257*(3), 845-858.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations.*

Wouda, N. A., Lan, L., & Kool, W. (2024). PyVRP: a high-performance VRP solver package. *INFORMS Journal on Computing*, *36*(4), 943–955.